

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-282667

(43)Date of publication of application : 15.10.1999

(51)Int.Cl.

G06F 9/06  
 G06F 12/14  
 G06F 15/78  
 G09C 1/00  
 H04L 9/14  
 H04L 9/32

(21)Application number : 10-103958

(71)Applicant : NAKAMICHI CORP

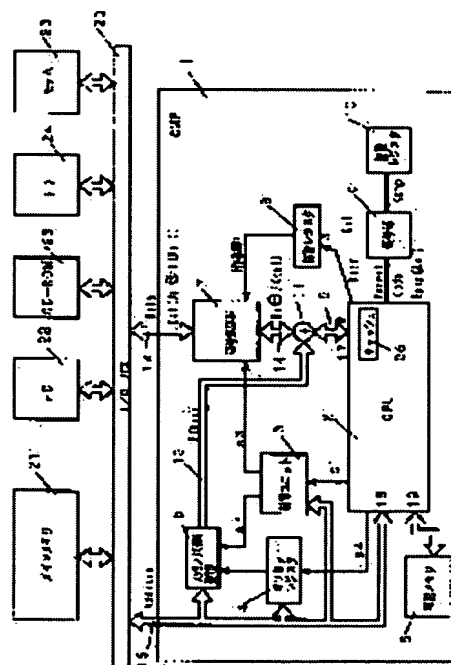
(22)Date of filing : 31.03.1998

(72)Inventor : SUEMATSU TOSHINARI

**(54) MICROPROCESSOR HAVING CIPHER PROCESSING FUNCTION OF MULTIPLE KEY SYSTEM****(57)Abstract:**

**PROBLEM TO BE SOLVED:** To obtain the data which can be decoded by only a CMP and to prevent the disadvantage caused by the leakage of data by enciphering and decoding a data group by means of a job key that is enciphered by a secret key proper to the CMP.

**SOLUTION:** A CMP 1 has its proper secret key  $K_{cmp}$  that is stored in its built-in storage register 10 and to be used for the decoding jobs of a decoding part 9. An offset register 4 receives a fetch signal  $s_4$  which is outputted from a CPU 2 and outputs the head address data to a scramble code generator 6. The generator 6 outputs a scramble code  $F(Re)$  generated from the head address data and the address data inputted from an address bus 16 to an adder 11 via a signal path 13. The adder 11 performs an exclusive OR operation between the code  $F(Re)$  and the data which are inputted from one of data paths 14 or 15. Then the adder 11 outputs the result of its exclusive OR operation to the other one of paths 14 and 15 to carry out the scramble and descramble operations.

**LEGAL STATUS**

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

**THIS PAGE BLANK (USPTO)**

(51) Int.Cl. <sup>8</sup>	識別記号	F I
G 0 6 F 9/06	5 5 0	G 0 6 F 9/06 5 5 0 A
		5 5 0 E
12/14	3 2 0	12/14 3 2 0 B
15/78	5 1 0	15/78 5 1 0 G
G 0 9 C 1/00	6 2 0	G 0 9 C 1/00 6 2 0 Z
審査請求 未請求 請求項の数 6 F D (全 20 頁) 最終頁に続く		

(21) 出願番号 特願平10-103958

(22) 出願日 平成10年(1998) 3月31日

(71) 出願人 000110468

ナカミチ株式会社

東京都小平市鈴木町 1 丁目153番地

(72) 発明者 末松 俊成

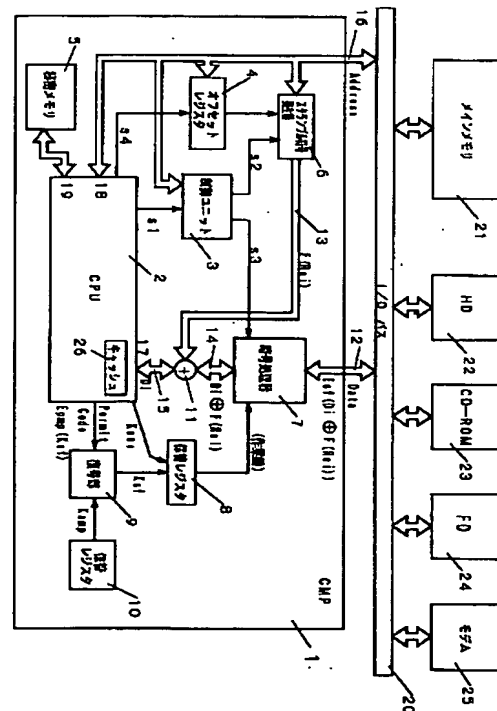
東京都小平市鈴木町 1 丁目153番地 ナカミチ株式会社内

## (54) 【発明の名称】 多重鍵方式の暗号処理機能を有するマイクロプロセッサ

## (57) 【要約】

【目的】 鍵で暗号化したソフトウェアを特定のコンピュータでのみ使用可能にし、且つこの暗号化ソフトウェアは、例えばCD-ROMのように同一の複製品を市場で販売可能とする。一方、暗号化ソフトウェアを復号する場合、暗号化したコンピュータでのみ復号化が可能とし、データの漏洩による不利益を防止する。

【構成】 ソフトウェアに用意された作業鍵によって暗号された暗号化ソフトウェアと、後述する秘密鍵に対応する公開鍵で前記作業鍵を暗号化した暗号コードとを別々に入手し、マイクロプロセッサに固有の秘密鍵を外に読み出し不可に保存する記憶手段と、この秘密鍵で前記暗号コードを復号化して前記作業鍵を生成する復号手段と、この作業鍵を保持する記憶手段と、この作業鍵によって前記暗号化ソフトウェアを復号化する復号手段とを有するマイクロプロセッサ。



1

## 【 特許請求の範囲】

【 請求項1 】 暗号化されたソフトウェアを復号化する第1 の復号手段と、

前記第1 の復号手段による復号化を行うための第1 の鍵を保存する第1 の記憶手段と、

暗号化された前記第1 の鍵を復号化して前記第1 の記憶手段に保存する第2 の復号手段と、

前記第2 の復号手段による復号化を行うための第2 の鍵を保存する第2 の記憶手段とを有することを特徴とする

1 チップで構成された多重鍵方式の暗号処理機能を有するマイクロプロセッサ。 10

【 請求項2 】 前記第1 の鍵と 前記第1 の記憶手段とをそれぞれ複数としたことを特徴とする請求項1 に記載の多重鍵方式の暗号処理機能を有するマイクロプロセッサ。

【 請求項3 】 前記第2 の鍵を復号化又は暗号化により生成することを特徴とする請求項1 に記載の多重鍵方式の暗号処理機能を有するマイクロプロセッサ。

【 請求項4 】 ソフトウェアに用意された第3 の鍵によって暗号化された暗号化ソフトウェアを復号化する1 チップで構成された多重鍵方式の暗号処理機能を有するマイクロプロセッサであって、 20

前記マイクロプロセッサ固有の第4 の鍵を該マイクロプロセッサ外に読み出し不可に保存する第3 の記憶手段と、

前記第4 の鍵に対応する第5 の鍵で前記第3 の鍵を暗号化した暗号コードを前記第4 の鍵で復号化して前記第3 の鍵を生成する第3 の復号手段と、

前記第3 の鍵を保持する第4 の記憶手段と、

前記第3 の鍵によって前記暗号化ソフトウェアを復号化する第4 の復号手段とを有することを特徴とする多重鍵方式の暗号処理機能を有するマイクロプロセッサ。 30

【 請求項5 】 ソフトウェアを暗号化／復号化する1 チップで構成された多重鍵方式の暗号処理機能を有するマイクロプロセッサであって、

第6 の鍵を暗号化した第7 の鍵を生成する第1 の暗号化手段と、

前記第7 の鍵を保存する第5 の記憶手段と、

前記第1 の暗号化手段による暗号化を行うための第8 の鍵を保存する第6 の記憶手段と、

前記第7 の鍵によって前記ソフトウェアを暗号化／復号化するための暗号化／復号化手段とを有することを特徴とする多重鍵方式の暗号処理機能を有するマイクロプロセッサ。 40

【 請求項6 】 ソフトウェアに用意された作業鍵によって暗号化された暗号化ソフトウェアを復号化する1 チップで構成された多重鍵方式の暗号処理機能を有するマイクロプロセッサであって、

前記マイクロプロセッサ固有の秘密鍵を該マイクロプロセッサ外に読み出し不可に保存する第7 の記憶手段と、

前記秘密鍵に対応する公開鍵で前記作業鍵を暗号化した 50

2

暗号コードを前記秘密鍵で復号化して前記作業鍵を生成する第5 の復号手段と、

前記作業鍵を保持する第8 の記憶手段と、

前記作業鍵によって前記暗号化ソフトウェアを復号化する第6 の復号手段とを有することを特徴とする多重鍵方式の暗号処理機能を有するマイクロプロセッサ。

## 【 発明の詳細な説明】

## 【 0 0 0 1 】

【 発明の属する技術分野】 本発明は、暗号処理機能を有するマイクロプロセッサに関し、特に暗号化されたソフトウェアを復号化する際に、複数の鍵を用いて行うマイクロプロセッサに関する。

## 【 0 0 0 2 】

【 従来の技術】 従来のこの種の暗号化方式に於いては、1 種類の鍵でソフトウェアの復号化を行う方式のコンピュータシステムが提案されている。

## 【 0 0 0 3 】

【 発明が解決しようとする課題】 従来方式の場合、同じ内容のソフトウェアであっても、コンピュータシステム毎に異なる鍵で暗号化するため、例えばCD-ROMのような同一の複数媒体を市場に投入出来ない。また、あるコンピュータシステムがソフトウェアを暗号／復号する場合、鍵が限定されるために解読されやすく、安全上に問題があった。

## 【 0 0 0 4 】

【 課題を解決するための手段】 ソフトウェアに用意された例えば作業鍵によって暗号化された暗号化ソフトウェアと、例えば後述する秘密鍵に対応する公開鍵で前記作業鍵を暗号化した暗号コードとを別々に入手し、例えばマイクロプロセッサに固有の秘密鍵を外に読み出し不可に保存する記憶手段と、この秘密鍵で前記暗号コードを復号化して前記作業鍵を生成する復号手段と、この作業鍵を保持する記憶手段と、この作業鍵によって前記暗号化ソフトウェアを復号化する復号手段とを有するマイクロプロセッサによって前記暗号化ソフトウェアを復号化する。 30

【 0 0 0 5 】 また、例えばユーザーが設定した作業鍵を暗号化し、暗号化された作業鍵を生成する暗号化手段と、暗号化された作業鍵を保存する記憶手段と、この暗号化を行うための例えばこのマイクロプロセッサに固有の秘密鍵を保存する記憶手段と、前記暗号化された作業鍵によって前記ソフトウェアを暗号化／復号化するための暗号化／復号化手段とを有するマイクロプロセッサによって前記ソフトウェアを暗号化／復号化する。

## 【 0 0 0 6 】

【 発明の実施の形態】 図1 は、本発明の一実施例を示す構成図で、図中1 は、暗号化機能を有するマイクロプロセッサ( 以下CMP と称す) であり、1 チップで構成されている。この中の2 は中央演算処理装置( 以下CPU と称す) を示し、3 は制御ユニットを示す。この制御ユ

ニット3は、CPU2からの指令信号s1を入力して、スクランブル符号発生器6の動作のオン、オフを指令する動作指令信号s2と暗号処理器7の暗号化／復号化機能の実行、停止の各動作を指令する動作指令信号s3とをそれぞれ出力する。

【0007】このCMP1は、固有の秘密鍵K<sub>sm</sub>を所有する。これは内蔵の保管レジスタ10に保管され、後述するごとくこの秘密鍵に対応する公開鍵K<sub>pm</sub>で作業鍵K<sub>sf</sub>を暗号化したパーミットコードを復号部9で復号化するための鍵となる。この秘密鍵K<sub>sm</sub>は、CMPの製造元のみが知るもので、CMPの外に現れないようにされてユーザー等の第3者が知ることは出来ず、逆に公開鍵K<sub>pm</sub>は、CMPの入手時にユーザーに知らされて全ての人が知り得るものである。ここで復号化された作業鍵K<sub>sf</sub>は、保管レジスタ8に保管される。

【0008】CMP1につながるアドレスバス16とデータバス12は、I/Oバス20を介してメインメモリ21、ハードディスク22、CD-ROM23、フロッピディスク24の各ドライブ、及びモデム25の各入出力ポートにつながっている。

【0009】一方CMP1内部において、アドレスバス16は、CPU2のアドレスポート18、制御ユニット3、オフセットレジスタ4、及びスクランブル符号発生器6につながり、データバス12は、暗号処理器7、加算回路11を介してCPU2のデータポート17につながっている。

【0010】オフセットレジスタ4は、CPU2から出力される取り込み信号s4を受信し、後述するようにCPU2がメインメモリ21との間で読み書きする際のデータの先頭位置を示す先頭アドレスデータを取り込んでメモリし、この先頭アドレスデータをスクランブル符号発生器6に出力する。スクランブル符号発生器6は、後述するようにこの先頭アドレスデータとアドレスバス16から入力するアドレスデータをもとに生成したスクランブル符号F(Re)を信号経路13を介して加算器11に出力する。

【0011】加算器11は、このスクランブル符号F(Re)とデータ経路14、15の一方のデータ経路から入力するデータの排他論理和演算を行い、その結果を他方のデータ経路に出力することによりこれ等のデータのスクランブルとデスクランブルを実行する。

【0012】ここで、図1の各部で、データに対して行う暗号化、復号化、及びデータスクランブルの各内容について更に説明する。ここで使用される共通鍵方式と公開鍵方式の2つの暗号化方式は一般的なものであり、その詳細な説明は省略するが、本文で用いるそれらの記述方法について説明する。

【0013】共通鍵方式では、データDを暗号化するときに秘密鍵K<sub>ab</sub>を用い、これによって暗号化されたデータをE<sub>ab</sub>(D)と標記する。この場合、秘密鍵K<sub>ab</sub>があ

れば、E<sub>ab</sub>(D)と記述された暗号化データを復号化して再びデータDを取り出すことができる。

【0014】一方、公開鍵方式では、データDを暗号化するときに、公にされた公開鍵K<sub>pab</sub>を用いて行ない、これによって暗号化された暗号化データをE<sub>pab</sub>(D)と標記する。この場合、この暗号化データの解読はこの公開鍵K<sub>pab</sub>に対応する秘密鍵K<sub>sab</sub>によってのみ行なうことができる。

【0015】即ち、共通鍵方式では、一つの秘密鍵K<sub>ab</sub>によってデータの暗号化と復号化を行なうが、公開鍵方式では、公にする公開鍵K<sub>pab</sub>ではデータの暗号化のみが可能であって、その復号化には秘密鍵K<sub>sab</sub>を用いて行なう。

【0016】次に、アドレスによって行なわれるデータスクランブルについて、その一例を示す本実施例の原理を説明する。簡単のため、データDが記憶手段に書き込まれる際には1バイト単位で行なわれ、各単位毎にアドレスA<sub>d</sub>を対応させて処理するものとする。また暗号化スクランブルされるデータは、メインメモリ21内の連続した領域におかれ、この一連の連続するデータの集合をデータ群と呼ぶことにする。1つのプログラムやデータは、通常1つ以上のデータ群で構成されるが、スクランブルは、データ群毎に処理される。

【0017】以降では説明を簡単にするため、プログラムやデータは、1データ群で構成されるものと仮定する。そこで、データ群DXを1バイトデータのデータ列、D1、D2……DNの集合として標記し、アドレス群A<sub>dX</sub>をこれ等の各1バイトデータに対応して付されるアドレスA<sub>d1</sub>、A<sub>d2</sub>……A<sub>dN</sub>の集合として標記する。

【0018】従って、この連続データの各アドレスA<sub>di</sub>(i=1,2,……N)は、その先頭アドレスをT<sub>o</sub>とし(通常はT<sub>o</sub>=A<sub>d1</sub>)、この先頭アドレスからの相対位置を示す相対アドレスをR<sub>ei</sub>としたとき、A<sub>di</sub>=T<sub>o</sub>+R<sub>ei</sub>として表せる。図1のオフセットレジスタ4は、この先頭アドレスT<sub>o</sub>をメモリするものである。

【0019】一方、スクランブル符号発生器6は、逐次入力するアドレスA<sub>di</sub>と先頭アドレスT<sub>o</sub>からこの相対アドレスR<sub>ei</sub>を算出し、この値に1対1で対応するも、規則性を示さない1バイトのスクランブル符号F(Re<sub>i</sub>)を発生する。このようにして生成されるスクランブル符号の集合をF(ReX)と標記する。加算器11は、データ経路14、15の内、一方から逐次入力する1バイトのデータと、このデータのアドレスに対応して入力するスクランブル符号F(Re<sub>i</sub>)との排他論理和データを他方のデータ経路に出力する。

【0020】従って、例えば、CPU2から出力されるデータD1がメインメモリ21にメモリされる時の一過程を説明すると、この時にデータD1、例えば(10010001)が出力されると共に、このデータの番地を指定するアドレスA<sub>d1</sub>が出力される。加算器11は、このデータを

10

20

30

40

50

5

入力すると共に、この時の指定アドレスから算出された相対アドレスR<sub>el</sub>に対応して発生したスクランブル符号F (Re<sub>l</sub>)、例えば(11001100)を入力し、これ等の排他論理和(01011101)を暗号処理器7に出力する。

【0021】このように、データD<sub>i</sub>とスクランブル符号F (Re<sub>i</sub>)との排他論理和によってスクランブルされたデータを( D<sub>i</sub>(+)F (Re<sub>i</sub>))と記述して図1中に示す。従って、この図からもわかるように、データ経路14はスクランブルされたデータの経路となる。

【0022】次に、スクランブルされたデータを解除する例として、CPU2が上記したデータD<sub>1</sub>を読み込む時の一過程を考えてみる。この時、このデータD<sub>1</sub>を読み出すためのデータとして上記アドレスA<sub>d1</sub>が出力され、これに伴ってスクランブル符号F (Re<sub>1</sub>)がスクランブル符号発生器6から出力されることが理解される。

【0023】従って加算器11は、この時データ経路14に表れるスクランブルされた( D<sub>1</sub>(+)F (Re<sub>1</sub>))のデータ列(01011101)とスクランブル符号F (Re<sub>1</sub>)のデータ列(11001100)との排他論理和(10010001)、即ちデスクランブルされる前の元のデータD<sub>1</sub>をCPU2に出力する。

【0024】このように、加算器11は、データD<sub>1</sub>とこれに対応するスクランブル符号F (Re<sub>1</sub>)との排他論理和をとってスクランブルされたデータ( D<sub>1</sub>(+)F (Re<sub>1</sub>))を形成し、逆にこのデータ( D<sub>1</sub>(+)F (Re<sub>1</sub>))とスクランブル符号F (Re<sub>1</sub>)との排他論理和をとってデスクランブルされる前の元のデータD<sub>1</sub>を生じるべく動作することが理解される。

【0025】次に、図2の実行準備フローに従って、暗号化され更にスクランブルされたアプリケーションソフトを入手し、その実行に際してCPU2内への読み込みを可能にするまでの準備過程について主に説明する。この時のアプリケーションソフトをデータ群D<sub>apX</sub>で示すと、上記した如くこのデータ群D<sub>apX</sub>に対応して生成されるスクランブル符号F (Re<sub>X</sub>)でスクランブルされ、共通鍵方式の作業鍵K<sub>sf2</sub>で暗号化されたアプリケーションソフトは、E<sub>sf2</sub>( D<sub>apX</sub>(+)F (Re<sub>X</sub>))で示される。

【0026】尚、この作業鍵K<sub>sf2</sub>は、前記した共通鍵方式の秘密鍵に相当するものであるが、公開鍵方式の秘密鍵と区別するために以後作業鍵と称することにする。また、ここで設定される作業鍵K<sub>sf2</sub>は、このアプリケーションソフト供給元がこのソフト専用に用意するもので、この作業鍵K<sub>sf2</sub>で他のアプリケーションソフトを解読することは出来ない。

【0027】この暗号化されたアプリケーションソフトE<sub>sf2</sub>( D<sub>apX</sub>(+)F (Re<sub>X</sub>))は、図10に示すように後述するパーミットコードを管理するための暗号化されていない起動処理プログラムと一対にされた配給ソフトとしてソフト供給元から供給される。

【0028】図示しない入力手段により所望のアプリケ

6

ーションソフトの実行指令を受けると(ステップ1)、CPU2は、先ずスクランブル符号発生器6の出力を停止すると共に、暗号処理器7の復号機能を停止するための指令信号s<sub>1</sub>を出力する(ステップ2)。そして次のステップ3で、この配給ソフトがメインメモリ21にロードされ、更に起動処理プログラムがCPU2に読み込まれて実行される。

【0029】この暗号化されたアプリケーションソフトE<sub>sf2</sub>( D<sub>apX</sub>(+)F (Re<sub>X</sub>))がメインメモリにロードされる際に、メインメモリ21に指定される先頭アドレスT<sub>o</sub>はその管理状況によって変わるが、前記したようにそれに続くアドレスは連続的に指定されるため、この時の先頭アドレスT<sub>o</sub>からの相対位置を示す相対アドレス群は、スクランブル時に用いられた相対アドレス群R<sub>eX</sub>と同じデータとなる。

【0030】この配給ソフトの供給手段としてはハードディスク22、CD-ROM23、フロッピーディスク24及び、モデム25等による方法が考えられるが、その方法は一般的に行なわれている方法を採用できるため、ここでその詳しい説明は省略する。

【0031】続くステップ4から10のステップは、いま実行された起動処理プログラムに従って実行され、次のステップ4では、この暗号化アプリケーションソフト専用の作業鍵K<sub>sf2</sub>を公開鍵K<sub>mp</sub>で暗号化したパーミットコードE<sub>mp</sub>( K<sub>sf2</sub>)がハードディスク22に既にメモリされているかを確認する。この公開鍵K<sub>mp</sub>は、上記したように図1のCMP1固有の秘密鍵K<sub>mp</sub>に対応するもので、この秘密鍵K<sub>mp</sub>のみによって復号することが可能となる。

【0032】従って、この場合のパーミットコードE<sub>mp</sub>( K<sub>sf2</sub>)は、公開鍵で作業鍵を暗号化したものであり、アプリケーションソフト供給元が、CMP1のユーザーから公開鍵K<sub>mp</sub>を入手して作成する。ステップ4でこのパーミットコードの存在が確認できないと、CMP1のユーザーは何かの方法でソフト供給元からこれ入手し(ステップ5)、このアプリケーションソフトの専用ファイルにパーミットコードを入力し、ハードディスク22にこれを保存する(ステップ6)。この様にパーミットコードの入力保存は、通常は暗号化されたアプリケーションソフトと対になったこの起動処理プログラムによって行なわれる。

【0033】次に、パーミットコードE<sub>mp</sub>( K<sub>sf2</sub>)の存在を確認するとこれを読み込み(ステップ7)、内蔵の保管レジスタ10に保管している秘密鍵K<sub>mp</sub>でこれを復号化して得た作業鍵K<sub>sf2</sub>を保管レジスタ8に保管する(ステップ9)。

【0034】次にCPU2は、スクランブル符号発生器6の出力を再開し、暗号処理器7の暗号化/復号化機能を起動するための指令信号s<sub>1</sub>を出力する(ステップ10)。そしてステップ11に至り、必要に応じてこのメ

7

インメモリ21に保管された暗号化されたアプリケーションソフトEsf2(DapX(+)F(ReX))がCPU2へ読み込まれて実行される。

【0035】従って、この時この暗号化されたアプリケーションソフトEsf2(DapX(+)F(ReX))は、暗号処理器7によって作業鍵Ksf2で復号化されて(DapX(+)F(ReX))となる。また上記のごとく、この時のアドレスデータから生成されるスクランブル符号はF(ReX)であり、加算器11は、これ等の排他論理和を行なってデスクランブルされたアプリケーションソフトDapXをCPU2のデータポート17に出力する。

【0036】以上のようにして、アドレススクランブルと暗号化されたアプリケーションソフトがCPU2に読み込まれる。このようにして暗号化されたアプリケーションソフトとパーミットコードの組合せにより、下記の長所をあげることができる。

【0037】長所

1. ソフト供給元は、供給するソフトを所定のCMPでのみ利用させることが出来る。CMPの秘密鍵Ksmがその外部に現れることがなく、ユーザーを含む第3者によってチェック出来ないため。

2. 暗号化されたアプリケーションソフトをCD-ROM等の複製可能な媒体により市場に出すことができる。同種のアプリケーションソフトは、同じ作業鍵で暗号化されるため。

3. 異種のアプリケーションソフトは、それぞれ専用の作業鍵で暗号化されなければならないが、アプリケーションソフトの種類だけ異なる作業鍵を用意すればよい。

【0038】次に、ユーザーによって作成され、一旦メインメモリ21に保存したデータを、暗号化して再度メインメモリ21に保存しなおす過程を図3のフローチャートを用いて説明する。この場合、このデータを作成する段階では、データの暗号化は行なわれないものとする。

【0039】このプログラムが起動すると、上記したようにユーザーによって作成されたデータ群DsXが暗号化されないままメインメモリ21に保管される(ステップ21)。次にCPU2は、キーボード等の入力手段(図示せず)から入力するユーザーからの指示に基づいてこのデータ群DsXを共通鍵方式で暗号化するための作業鍵Kusrを決定し(ステップ22)、これを保管レジスタ8に出力して保管する。

【0040】次にCPU2は、メインメモリ21のデータ群DsXを読み込むために、スクランブル符号発生器6の出力を停止すると共に、暗号処理器7の機能を一時停止するための指令信号s1を出力する(ステップ25)。この時データ群DsXはそのままCPU2に読み込まれるが1バイト毎にアドレス指定されて読み込まれる(ステップ26)、この段階では最初の1バイト分のデータDs1がCPU2に読み込まれる。尚、データ群Ds

8

Xは、1バイトデータDs1, Ds2, ……DsNの集合として標記する。

【0041】次にCPU2は、スクランブル符号発生器6の出力を再開し、暗号処理器7の復号化機能を起動するための指令信号s1を出力する(ステップ27)。この状態でCPU2は、取り込んだ1バイト分のデータDs1を再度メインメモリ21に書き込む。

【0042】この書き込み過程で、先ずこのデータの番地を指定するアドレスに対応してスクランブル符号発生器6から出力されるスクランブル符号F(Re1)と、このデータDs1との排他論理和(Ds1(+)F(Re1))によるアドレススクランブルが行なわれ、これがデータ経路14に現れる(ステップ28)。尚、スクランブル符号F(Re1)は、上記したデータ群DsXの各1バイトデータに対応して生成されるスクランブル符号F(Re1), F(Re2), ……F(ReN)の1番目に相当し、これ等の集合をF(ReX)と標記する。

【0043】更にこのデータ(Ds1(+)F(Re1))は、暗号処理器7によって作業鍵Kusrで暗号化され(ステップ28)、Eusr(Ds1(+)F(Re1))の状態ではメインメモリ21に書き込まれる。次に、再びステップ24に戻って、暗号化するデータ群DsXが終了するまで1バイトづつ同様の暗号化処理が繰り返される。

【0044】そして、処理するデータDsXがなくなった段階でステップ30に至り、スクランブル符号発生器6の出力停止と、暗号処理器7の暗号化機能停止のための指令信号s1を出力する。そしてこのメインメモリ21に書き込まれた暗号化されたデータを、HD, FD等の補助記憶装置に保存して(ステップ31)このプログラムを終了する。

【0045】尚、CPU2のデータポート17部にキャッシュメモリ26を設けると、入出力するデータをためることができ、暗号化処理効率を上げることが出来る。即ち、ステップ26で複数バイト、例えば100バイト分のデータを連続して読み込み、ステップ28でこの100バイト分のデータを連続してスクランブル及び暗号化することによって、ステップ25とステップ27の各機能の切り換え行程をその分省略することが出来る。

【0046】上記の例では、ユーザーが作成したデータを暗号化する行程を説明したが、前記したアプリケーションソフト提供元が、暗号化プログラムを作成する場合も同様の行程で処理することが出来る。アプリケーションソフトを暗号化する場合には、作成したアプリケーションソフトをデータ群DapXとしてメインメモリ21に一旦保管し、共通鍵方式の秘密鍵として作業鍵Ksf2を設定し、上記した図3のフローに従って処理すれば、前記した暗号化されたアプリケーションソフトEsf2(DapX(+)F(ReX))を補助記憶装置に保管することが出来る。

【0047】また上記の実施例の場合、ステップ2で一

且データをメインメモリに作成し、その後暗号化処理を行なうケースを示したが、データを作成する過程で直接暗号化してメモリに出力する方法も考えられる。この場合、作成したデータをメインメモリに出力する際に、その過程で1バイト毎にデータのアドレススクランブルと暗号化を行ない、メインメモリに保存するように構成すればよい。

【0048】また前記実施例では、オフセットレジスタ4にアドレス群の先頭アドレスをメモリするように説明したが、これに限定されるものではなく、相対アドレス  $R_{ei}$  を導き出せる基準値であればよい。

【0049】次に、CMP1内に設けられた秘密メモリ5の働きについて説明する。このため、ソフト供給元によってその稼働時間が管理されたアプリケーションソフトを、例えば図4乃至図7の各フローチャートに従ってCD-ROM23から取り込み、実行する場合を例に説明する。

【0050】これ等のフローチャートの説明の前に、理解を容易にするために秘密メモリ5の働きについて、その要点を先ず説明する。図9は、秘密メモリ5に保存されるデータの内容を示すが、ここには、CMP1が実行するn種類のアプリケーションソフトに1対1で対応するパスワードPwが保存される。

【0051】このパスワードPwは、秘密メモリ5内で唯一の値であればどのような値でも良く、後述する図4のインストールフローのステップ75では、生成される乱数によって決定される例を示す。この様にして決定されるパスワードPwは、秘密メモリ5内に保存されると共に、インストール実行中のアプリケーションソフトの環境設定ファイルにも保存され、後述する図5の実行フローのステップ47で秘密値  $N_{ra}$  の照合を行なう際に利用される。

【0052】この秘密値  $N_{ra}$  は、同じく図6の実行フローのステップ57でアプリケーションソフトの実行が終了する毎に任意に発生される乱数などによって更新され、稼働可能時間を示す残量時間  $T_{re}$  と共に実行中のアプリケーションソフトの時間データファイルDdat ( $Ddat = T_{re} + N_{ra}$ ) としてハードディスク22に保存される(ステップ59)。更に、ここで更新された秘密値  $N_{ra}$  は、ステップ60で秘密メモリ5内に確保された実行中のアプリケーションソフトに対応するパスワードPwによって同メモリ内で識別される秘密値  $N_{ra}$  に上書きされる。尚、図5、6に示す実行フロー内において、秘密値の更新前後を区別するために、更新前の秘密値を  $N_{ra}$  で、また更新後の秘密値を  $N_{ra}'$  としてそれぞれを示す。

【0053】このステップ60での秘密値の更新に際しては、後述するように安全性を高めるために秘密メモリ5の更新前の秘密値  $N_{ra}$  による照合が行なわれる。よって、更新時には(Pw,  $N_{ra}$ ,  $N_{ra}'$ ) が用意され、パス

ワードPwによって識別された秘密メモリ5内の  $N_{ra}$  と照合のために用意した  $N_{ra}$  とが一致した場合のみ、新しい秘密値  $N_{ra}'$  によって書換が行なわれる。このようにしてユーザー等の第3者による  $N_{ra}$  の安易な変更を防止する。

【0054】以上の様に秘密値  $N_{ra}$  を秘密メモリ5に保管すると共に、時間データファイルDdatとしてハードディスク22内に保存し、このアプリケーションソフトが実行される毎にこれ等を照合することによって、ハードディスク22に保管されている時間データファイルDdatが第3者によって改ざんされていないかをチェックすることができる。

【0055】例えばユーザーが、残量時間  $T_{re}$  が十分残っている段階で時間データファイルDdatをコピーして予め別途確保しておき、このアプリケーションソフトが実行されて残量時間が僅かになった段階で、ハードディスク22に保存される時間データファイルDdatと置き換えたとする。しかしながらこの場合、この置き換えられた時間データファイルの秘密値と秘密メモリ5に保存してある秘密値を照合することによって、このデータが正規のデータでないことが判明する。なぜならば、この場合の両者の秘密値  $N_{ra}$  は、生成されたタイミングが異なるため、当然異なった値となっているからである。

【0056】以上の記載から理解されるように、秘密メモリ5内のデータは、ユーザーを含め第3者によって操作出来ないようにすることが好ましい。従って、CMP1の外部に秘密メモリ5のデータが出力されないように構成される必要がある。

【0057】図4乃至図7の各フローチャートは、上記したように、不正なデータの改ざんを防ぐべく、秘密メモリ5を使って実施される秘密データ管理方法の全体の手順を示すもので、以下その流れを順に説明する。

【0058】先ず、所望のアプリケーションソフトを入手してハードディスク22にインストールし、更に必要な諸環境を初期設定するまでの流れを図4のインストールフローに従って説明する。所望のアプリケーションソフトの入手方法は種々考えられるが、このソフトが入ったCD-ROMを所定のルートで入手し(ステップ71)、インストールする例を記述する。尚、このインストールフローは、アプリケーションソフトに付随するセットアッププログラムファイルに基づいて実行されるものである。

【0059】この場合、このCD-ROM23がCD-ROMドライブにセットされ、図示しない入力手段によってインストールの指示を受けると、そのアプリケーションソフトがハードディスク22にインストールされる(ステップ72)。他のインストール例として、アプリケーションソフトをモデム25を介してダウンロードし、ハードディスク22にインストールするようにしてもよい。

10

20

30

40

50

## II

## 12

【0060】但し、この時インストールされたアプリケーションソフトDexeXは、前記した方法でスクランブル符号F(ReX)によるアドレススクランブルと作業鍵Ksf3による暗号化が行なわれ、Esf3(DexeX(+F(ReX)))の状態ハードディスク22に保存されているものとする。

【0061】そして次のステップ73では、作業鍵Ksf3を確保し、保管レジスタ8にこれを保管する。そのため、このステップでは、図2のフローのステップ5からステップ9で説明したように所定の手順でソフト供給元から入手したパーミットコードEpm(Ksf3)をハードディスク22に保管し、復号部9で復号化する作業が含まれる。

【0062】そして次のステップ74では、スクランブル符号発生器6の出力を開始し、暗号処理器7の暗号化／復号化機能を起動するための指令信号s1を出力する。これにより、後述するステップ82で両機能がオフとされるまで、CMP1とメインメモリ21等の外部記憶装置との間で入出力するデータはすべて暗号化されたものとなる。

【0063】次のステップ75では、前記したようにこのアプリケーションソフトに1対1で対応するパスワードPw3を生成すべく乱数を発生させ、まだ秘密メモリ5内に存在しない数値であることを確認して確定し、同メモリ内にこれを確保する。そして、インストールされたアプリケーションソフトDexeXの環境設定ファイルAcf3を作成し、この中に生成したパスワードPw3を書き込んでハードディスク22に保存する(ステップ76)。

【0064】次に、このアプリケーションソフトの稼働可能な残量時間Treをゼロとし(ステップ77)、乱数を発生させて秘密値Nraを生成する(ステップ78)。そして残量時間Treと秘密値Nraとからなる時間データファイルDdat(Ddat=Tre+Nra)を作成し、ハードディスク22にこれを保存する(ステップ79、80)。

尚、この時作成される時間データファイルDdatは、必要に応じて一旦メインメモリ21に保存されるが、この際のデータは当然暗号化され、且つスクランブルされたデータとして保存される。

【0065】また、この時生成した秘密値Nraは、図9に示すように秘密メモリ5内に確保されたパスワードPw3に対応してこれに保存され(ステップ81)、このパスワードPw3によって照合などの操作が可能とされる。そして、ステップ82でスクランブル符号発生器6の出力と、暗号処理器7の暗号化／復号化機能と停止してこのフローを終了する。

【0066】以上のステップによって所望のアプリケーションソフトDexeXが暗号化された状態でハードディスク22にインストールされ、更に必要な諸環境が初期設定されたことになる。

【0067】次に、所望のアプリケーションソフトを実行する場合の手順について、図5、6の実行フローチャートを参照しながら説明する。説明を容易にするため、ここで実行するアプリケーションソフトを、上記の説明でインストールされたDexeXとする。

【0068】ステップ41でこのアプリケーションソフトが実行されると、このソフトの読み込みが行なわれる(ステップ42)。このステップ42の行程は、前記した図2の示すフローチャートのステップ2からステップ10の行程に対応するもので、既に説明したように暗号化スクランブルされたアプリケーションソフトEsf3(DexeX(+F(ReX)))がメインメモリ21にロードされると共に、パーミットコードEpm(Ksf3)が復号され、作業鍵Ksf3が保管レジスタ8に保管される。

【0069】更に、スクランブル符号発生器6の出力を開始し、暗号処理器7の暗号化／復号化機能を起動するための指令信号s1を出力する。これにより、後述するステップ49で両機能がオフとされるまで、CMP1とメインメモリ21等の外部記憶装置との間で入出力するデータはすべて暗号化されたものとなる。

【0070】尚、図2で説明したように、この暗号化スクランブルされたアプリケーションソフトEsf3(DexeX(+F(ReX)))には前記した起動処理プログラムが添付されているものとする。また今回は、図4のフローに基づくインストール時にパーミットコードの保管が行なわれるので、結果的に図2中の起動処理プログラムによるステップ4からステップ6の行程を省略できる。

【0071】そして、このソフトの環境設定ファイルAcf3を読み出し、この中にパスワードPwがあるか確認する(ステップ43)。このケースでは、パスワードPw3が存在するのでステップ44に至るが、もし存在しない場合、このソフトのインストール時にエラーが生じてパスワードが生成されなかったことを意味するので、このことを図示しない表示手段で表示して再インストールの指令を出し(ステップ45、46)、更にステップ49でスクランブル符号発生器6の出力と暗号処理器7の機能とを停止した後、待機状態に戻る。

【0072】次にステップ44でこのアプリケーションソフトの時間データファイルDdat(Ddat=Tre+Nra)が読み出される。この時、当然作業鍵Ksf3による復号化とアドレスによるデスクランブルが行なわれる。尚、この時読み出される暗号化された時間データファイルDdatは、必要に応じてメインメモリ21にそのままの形で保管される。

【0073】次に、秘密メモリ5内の秘密値Nraと時間データファイルDdatの秘密値との照合が行なわれる(ステップ47)。今は、上記図4のインストールフローの説明でインストールされたアプリケーションソフトDexeXの秘密値が照合されるため、この時間データファイルDdatが何らかの方法で改ざんされないかぎりこ

れら2つの秘密値は一致し、ステップ50に移行することが理解される。

【0074】もしここで、前記したような時間データファイルDdatの置き換えが行なわれていると、当然これら2つの秘密値が異なるためステップ48に移行し、後述する処理が行なわれる。

【0075】ステップ50乃至ステップ53は、アプリケーションソフトDexeXを実行処理すると同時に残量時間を監視するフローになっている。即ち、ステップ50では実行時間を計測して逐次残量時間を更新し、ステ

ップ51では残量時間がゼロになったかを監視し、更にステップ53ではこの実行の停止指令が入ったかどうかを監視する。

【0076】また、このステップ52では、必要に応じて順次メインメモリ21にロードされたアプリケーションソフトEsf3(DexeX(+)F(ReX))がCPU2へ読み込まれるが、その際には前記した復号化、及びデスクランブルが逐次実行される。

【0077】この実行中に残量時間がゼロになるとステップ51でこれを判定し、一旦ソフト処理を終了した後(ステップ54)、図示しない表示手段によって例えば「残量時間がありません。」等の表示を行なって(ステップ55)、後述するようにソフト供給元に残量時間の追加を申請するプログラムに入る(ステップ56)。今は、インストールされたばかりのアプリケーションソフトDexeXの例を想定して記述しているので残量時間がゼロになっており、当然このルートをとる。

【0078】このステップ56の残量時間追加申請プログラムは、図7に示すフローチャートに示す手順によって行なわれる。即ちユーザーは、残量時間がなくなって暗号化された状態の時間データファイルDdatを何等かの方法でソフト供給元に届ける(ステップ91)。その方法は種々考えられるが、例えばモデム25を介して行なわれるインターネットなどのパソコン通信によって行なってもよい。

【0079】一方、ソフト供給元では、このファイルの復号化とデスクランブルを行なってデータを復元し、このDdat(Ddat=Tre+Nra)の残量時間Treの追加補

充を行なう(ステップ92)。その補充量は、ユーザーの依頼に応じて決定されるが、対価の補充料金の支払については別途行なわれるものとし、ここでの説明は省略する。

【0080】ソフト供給元では、この残量時間Treの補充を行なった後、一緒に送られてきた秘密値と共に、再びアドレススクランブルと作業鍵Ksf3による暗号化を行なった後ユーザーに返信する(ステップ93)。この方法も、パソコン通信を介して行なってもよいし、フロッピーディスクに納めて郵送するようにしてもよい。

【0081】ユーザーは、この時間補充された時間データファイルを再度入手し、前のデータファイルと置き換

える如くこれをハードディスク22に保存し(ステップ94)、この残量時間追加申請のフローを終了する。従って、この時のDdat(Ddat=Tre+Nra)において、残量時間Treは所定量補充されているものの、秘密値Nraは、秘密メモリ5に保存されている秘密値と同じままであることが理解される。

【0082】ユーザーは、このアプリケーションソフトDexeXを引き続いて実行したい場合、再度図5の実行フローのステップ41に戻らなければならない。然し乍ら今回は、残量時間が所定量補充されているので、ステップ41で実行が指令されると、ステップ50乃至ステップ53のフローが繰り返されて、アプリケーションソフトDexeXの実行状態が継続されることになる。

【0083】そして、残量時間Treがゼロとなる前に、ユーザーによるソフト処理終了の指令を受けると、次のステップ57に至る。このステップ57では、乱数を発生して秘密値Nraを更新し、その後この更新した秘密値Nra'とこの時のソフトの実行時間が差し引かれた残量時間Treとを対にして新たな時間データファイルDdat(Ddat=Tre+Nra')を作成する(ステップ58)。尚、この時作成されたデータDdatは、必要に応じて一旦メインメモリ21に保存されるが、この際のデータは当然暗号化され、スクランブルされたデータとして保存される。

【0084】そしてこれをハードディスク22に保存されている更新前の時間データファイルに代えて保存する(ステップ59)と共に、更新された秘密値Nra'を、秘密メモリ5内に確保されたパスワードPw3によって識別可能に同メモリ内に保持されている更新前の秘密値に代えて保存する(ステップ60)。

【0085】この時、前記した更新前の秘密値Nraの照合が行なわれる。図11に示すフローチャートは、ステップ60内で行なわれるこの照合動作の流れを示すもので、ステップ60-1では秘密メモリ5内の秘密値Nraと照合のために別途用意されたNra'とが合致するかをチェックする。これ等が合致する場合はステップ60-2で上記したデータの更新が行なわれ、合致しない場合は、ステップ60-3に至って、図示しない表示手段によって、「不正が行なわれました。」等の表示を行ない、図5のBへ戻るように構成しても良い。

【0086】そして再度このアプリケーションソフトDexe3が実行されると、上記した各ステップを経てステップ47に至り、ここで秘密値が照合される。この時、秘密メモリから読み出した秘密値とハードディスク22から読み出された秘密値とは一致するはずであるが、もし、ユーザーによって前記したようなデータの改ざんがあるとこれ等は一致しない。この時はステップ48に至って「不正行為が行なわれました。」等のメッセージを表示し、実行を中止すべくステップ49を経由して待機状態に戻る。

15

【0087】一方、このアプリケーションソフトDexeXの使用を中止し、アンインストールする場合には、このソフトに設定されたパスワードPw3と秘密値Nraを秘密メモリ5から消去する必要がある。これを行なわないと同メモリ内に不要な秘密値が蓄積されることになる。またこの消去作業はアンインストール時のみに、またハードディスク22にパスワードが存在する段階で行なわれる必要があるため、例えば、図8に示すフローに従って行なわれる。

【0088】即ち、図示しない入力手段によってアンインストールの指示を受けると、ステップ101に至り、先ずパスワードと秘密値の消去が行なわれる。この際にも、安全性を高めるために前記したような秘密値Nraによる照合が行なわれる。このために(Pw3, Nra)が用意され、パスワードPw3によって識別された秘密メモリ5内のNraと照合のために用意したNraとが一致した場合にのみこのアプリケーションソフトDexeXに対応するパスワードPw3と秘密値Nraとが同時に消去される。そしてこの消去が終了した後、アプリケーションソフトDexeX自体のアンインストールが実行される(ステップ102)。

【0089】次に、本発明の他の実施例について説明する。図14は、秘密メモリ部51がCMP50の外にあって独立して存在する場合を示し、その他の前記した図1と同じ機能を有する構成要素については、同符号を付してその説明を省略する。図1に対する図14の構成上の違いは、CMP1内部にあった秘密メモリがその外部の秘密メモリ部51に照合手段52と共に独立して設けられた点であり、全体的な機能は全く同じである。

【0090】一方、図1に対する図14の部分的な機能の相違点は、図1の構成では秘密メモリ5の内部データがCMP1の外部に出力されないように構成されているのに対し、図14の構成では秘密メモリ部51の内部データがその外部に出力されないように構成される点である。

【0091】従って、図1の構成のCMP1内に設けられた秘密メモリの働きについて、図4乃至図7のフローチャートを参照して説明したが、図14構成のCMP50外に設けられた秘密メモリ部51についても、これ等のフローチャートに従って同様の作業が実行できる。しかしながら、図5のステップ47、図11のステップ61-1、及び図8のステップ101の各ステップで行なわれる照合作業は、秘密メモリ部51内部の照合手段52で行なわれ、YES,NOの結果のみが出力されるように構成されている。

【0092】次に、本発明の他の実施例について説明する。図12は、本発明の他の実施例を示す構成図で、前記した図1と同じ機能を有する構成要素については、同符号を付してその説明を省略し、図1の構成及び動作と異なる部分について重点的に説明する。

16

【0093】この実施例の場合、暗号化されアドレススクランブルされたデータDを前記した標記方法に従って標記すると、Escri(D)となる( $i=1,2,3\cdots N$ )。但しこの秘密鍵Kscriは、 $Kscri=Ksf(+ )F(Rei)$ ということになる。即ち、このデータDに用意される作業鍵Ksfと逐次生成されるスクランブル符号F(Rei)との排他論理和符号によってデータDを暗号化したものとなる。

【0094】このようにして暗号化されたアプリケーションソフトを図12のCMPによって復号化する過程を説明する。いま、このアプリケーションソフトの1バイト単位のデータ群D1, D2 $\cdots$ DNの集合をDXと記述し、これに対応して生成されるスクランブル符号群F(Re1), F(Re2) $\cdots$ F(ReN)の集合をF(ReX)と記述し、この暗号化されたアプリケーションソフトをEscr(DX)と記述する。但し、 $Kscr=Ksf(+ )F(ReX)$ である。

【0095】従って、いま最初の1バイトのデータD1を復号する過程を記述すると、この時データバス12には、暗号化されたデータEscr1(D1)が現れ、スクランブル符号発生器6からはF(Re1)が出力される。従ってこの時の暗号処理器7は、作業鍵Ksfとスクランブル符号F(Re1)の排他論理和符号Ksf(+ )F(Re1)によってデータEscr1(D1)を復号することになるため、暗号化した時と同じ鍵でこれを復号化して得たデータD1をCPU2に出力することが出来る。以下同様にして、全ての暗号化されたデータ群を順次復号化してCPU2に送る。

【0096】先に、図1に示す構成のCMP1による、アプリケーションソフトの入手、及びその実行に際してのCPU2内への読み込み過程について、図2のフローチャートを参照しながら説明したが、図12構成のCMP30もこのフローチャートに従って同様の作業が実行できる。但し、ステップ11の復号方法が上記した説明に基づいて実行されるところが異なる。

【0097】また、ユーザーによって作成され、一旦メインメモリ21に保存したデータを、図1の構成のCMPによって暗号化して再度メインメモリに保存しなおす過程を図3のフローチャートを参照して説明したが、図12構成のCMP30もこのフローチャートに従って同様の作業が実行できる。但し、ステップ28の暗号化方法が異なり、排他論理和符号( $Kusr(+ )F(ReX)$ )によってデータ群DXを逐次暗号化するものとなる。

【0098】更に、図1の構成のCMP1内に設けられた秘密メモリの働きについて、図4乃至図7のフローチャートを参照して説明したが、図12構成のCMP30内に設けられた秘密メモリ5についても、これ等のフローチャートに従って同様の作業が実行できる。但し、フローで実行されるデータの暗号化又は復号化の方法が図1のCMP1と異り、上記したように排他論理和符号( $Kusr(+ )F(ReX)$ )を作業鍵として行なわれる。

17

【0099】次に、本発明の他の実施例について説明する。図13は、本発明の他の実施例を示す構成図で、前記した図1と同じ機能を有する構成要素については、同符号を付してその説明を省略し、図1の構成及び動作と異なる部分について重点的に説明する。

【0100】この実施例の場合、暗号化されアドレススクランブルされたデータDを前記した標記方法に従って標記すると、 $E_{sss}(D(+)F(Re))$ となる。但しこの作業鍵 $K_{sss}$ は、 $K_{sss}=E_{ssp}(K_{usr})$ ということになる。即ち暗号部41は、CPU2からユーザーによって設定される作業鍵 $K_{usr}$ を入力し、保管レジスタ10が保管しているこのCMP40に固有の秘密鍵 $K_{ssp}$ でこれを暗号化した前記作業鍵 $K_{sss}$ を生成し、これを保管レジスタ8に出力する。

【0101】暗号処理部7は、この保管レジスタ8に保管された作業鍵 $K_{sss}$ でアドレススクランブルされたデータ $(D(+)F(Re))$ を暗号化し、この暗号化したデータ $E_{sss}(D(+)F(Re))$ をデータバス12を介してメインメモリ21等の外部メモリに出力する。

【0102】このようにして暗号化されたデータを図13のCMPによって復号化する過程を説明する。いま、データの1バイト単位のデータ群 $D1, D2, \dots, DN$ の集合をDXと記述し、これに対応して生成されるスクランブル符号群 $F(Re1), F(Re2), \dots, F(ReN)$ の集合を $F(ReX)$ と記述し、このスクランブル暗号化されたデータを $E_{sss}(DX(+)F(ReX))$ と記述する。但し、 $K_{sss}=E_{ssp}(K_{usr})$ である。

【0103】従って、いま最初の1バイトのデータD1を復号する過程を記述すると、この時データバス12には、暗号化されたデータ $E_{sss}(D1(+)F(Re1))$ が現れ、スクランブル符号発生器6からは $F(Re1)$ が出力される。従って暗号処理部7は、この時作業鍵 $K_{sss}$ でデータ $E_{sss}(D1(+)F(Re1))$ を復号化した $(D1(+)F(Re1))$ を加算器11に出力し、加算器11は $F(Re1)$ でこれをデスクランブルして得たデータD1をCPU2に出力することが出来る。以下同様にして、全ての暗号化されたデータ群を順次復号化してCPU2に送る。

【0104】また、ユーザーによって作成され、一旦メインメモリ21に保存したデータを、図1の構成のCMPによって暗号化して再度メインメモリに保存しなおす過程を図3のフローチャートを参照して説明したが、図13構成のCMPもこのフローチャートに従って同様の作業が実行できる。但し、ステップ23では作業鍵 $K_{sss}=E_{ssp}(K_{usr})$ を保管レジスタ8に保管する必要がある、このため前記した暗号部41より作業鍵 $K_{usr}$ の暗号化が行なわれる。

【0105】従って、この図13に構成されたCMP40によって暗号化されたデータは、この暗号化したCMPのみによって復号化されることが理解される。更に、作業鍵 $K_{sss}$ が二重鍵の構成を有するため、暗号化する

18

データ群ごとに作業鍵を変えることができるために鍵自体が解読されにくくなり、データの氾濫を防ぐと共にその機密性を高めることが出来る。

【0106】次に、本発明の他の実施例について説明する。図15は、本発明の他の実施例を示す構成図で、前記した図1と同じ機能を有する構成要素については、同符号を付してその説明を省略し、図1の構成及び動作と異なる部分について重点的に説明する。

【0107】この実施例の場合、作業鍵の保管レジスタ8を複数P個有する保管レジスタ部61を設け、これ等の各保管レジスタに異なる作業鍵 $K_{sfi}(i=1,2,\dots,P)$ をそれぞれ保管可能とする。この保管レジスタ部61は、復号部9から出力される作業鍵を保管するレジスタの選択、及び暗号処理部7に出力する作業鍵の選択を、制御ユニット3から出力される鍵選択信号s5によって実行できるように構成されている。

【0108】また、オフセットレジスタ部62には、前記したデータ群の先頭アドレスを保管するための複数P個のオフセットレジスタ4が、異なる先頭アドレス $T_{oi}(i=1,2,\dots,P)$ をそれぞれ保管可能に設けられ、制御ユニット3から出力される先頭アドレス選択信号s6によって、先頭アドレス $T_{oi}$ の保存、及び読み出しの対象となるレジスタが選択可能に構成されている。

【0109】次にこのような構成のCMP60の動作について説明する。今、前記した図2の実行準備フローが実行され、そのステップ3の段階でメインメモリ21にロードされたソフトが図17に示すようにM個のプログラムから構成され、それぞれが異なる作業鍵 $K_{sfj}(j=1,2,\dots,M, M \leq P)$ で暗号化されているものとする。

【0110】また各プログラムは、前記したようにそれぞれ所定単位のデータの集合であるデータ群で構成され、各データ群の先頭アドレスを $T_{oj}(j=1,2,\dots,M)$ と記す。この場合の各作業鍵 $K_{sfj}$ は、各プログラムを管理するソフト供給元が前記したパーミットコードとして供給するため、図2の実行準備フローのステップ4からステップ9までのステップが図1のCMP1と異なる。

【0111】即ち、ステップ4からステップ6までで、パーミットコードの存在の確認、入手、保存の各行程を実施するが、今このパーミットコードがM個必要となるため、この各行程もM回繰り返すように構成される。更に、ステップ7からステップ9までの行程で、パーミットコードを読み込み、それを復号し、生成した作業鍵をレジスタに保存するが、ここでもこの各行程をM回繰り返すように構成される。

【0112】この時逐次生成される作業鍵 $K_{sfj}$ が、CPUから出力される指令信号s1に基づいて制御ユニット3から出力される鍵選択信号s5によって選択された1からMまでの各保管レジスタ8にそれぞれ保管される。

10

20

30

40

50

【 0 1 1 3 】更にステップ11で、必要に応じてメインメモリ21に保管された各プログラムがCPU2に読み込まれて実行される際には、それぞれのプログラムの作業鍵Ksfjを保管するレジスタが選択され読み出され、暗号処理器7に供給される。これと共に、読み出されるプログラムに対応する先頭アドレスがCPUから出力される指令信号s1に基づいて制御ユニット3から出力されるアドレス選択信号s6によって選択される。そして選択された先頭アドレスTojがスクランブル符号発生器6に出力され、前記したアドレススクランブルが各プログラム毎に実行される。尚、iとjとが対応して管理されることにより、各プログラムとこれを実行する際に読み出される各作業鍵、及び各先頭アドレスとの対応関係が維持される。

【 0 1 1 4 】次に、本発明の他の実施例について説明する。図16は、本発明の他の実施例を示す構成図で、前記した図1と同じ機能を有する構成要素については、同符号を付してその説明を省略し、図1の構成及び動作と異なる部分について重点的に説明する。

【 0 1 1 5 】この実施例の場合、復号部72と保管レジスタ73とからなる多重鍵生成部71が追加構成される。そして保管レジスタ73は、このCMP70に固有の秘密鍵Ksmpを保管し、復号部72は前記した公開鍵Kpmpに対応する秘密鍵Ksmpを、秘密鍵Ksmpに対応する公開鍵Kpmpで暗号化したE pmp( K smp)を得て秘密鍵Ksmpで復号化し、生成した秘密鍵Ksmpを保管レジスタ10に出力する。

【 0 1 1 6 】このように構成することにより、CMP70にとって、一対の公開鍵Kpmpと秘密鍵Ksmpとがすでに登録された固有のものではなく、例えばCMP70の供給元から別途これ等の鍵を入手して設定することが出来るなど、種々の態様を可能とするものとなる。

【 0 1 1 7 】

【 発明の効果】本発明によれば、ソフト供給元は、供給するソフトを所定のCMPでのみ利用させることが可能となるだけでなく、同一のアプリケーションソフトは同じ鍵で暗号化されるため、暗号化したアプリケーションソフトをCD-ROM等の複製のみ可能な媒体によって市場に出すことができる。一方、図13の構成の本発明によれば、CMP固有の秘密鍵で暗号化された作業鍵によってデータ群を暗号化／復号化するため、同じCMPでしか解読出来ないデータとなり、データの漏洩による不利益を防ぐことが出来る。

【 図面の簡単な説明】

【 図1 】本発明の一実施例を示す構成図

【 図2 】本発明の動作説明に供するフローチャート

【 図3 】本発明の動作説明に供するフローチャート

【 図4 】本発明の動作説明に供するフローチャート

【 図5 】本発明の動作説明に供するフローチャート

【 図6 】本発明の動作説明に供するフローチャート

【 図7 】本発明の動作説明に供するフローチャート

【 図8 】本発明の動作説明に供するフローチャート

【 図9 】本発明の動作説明に供する秘密メモリの説明図

【 図10 】本発明の動作説明に供するアプリケーションソフトの構成図

【 図11 】本発明の動作説明に供するフローチャート

【 図12 】本発明の他の一実施例を示す構成図

【 図13 】本発明の他の一実施例を示す構成図

【 図14 】本発明の他の一実施例を示す構成図

【 図15 】本発明の他の一実施例を示す構成図

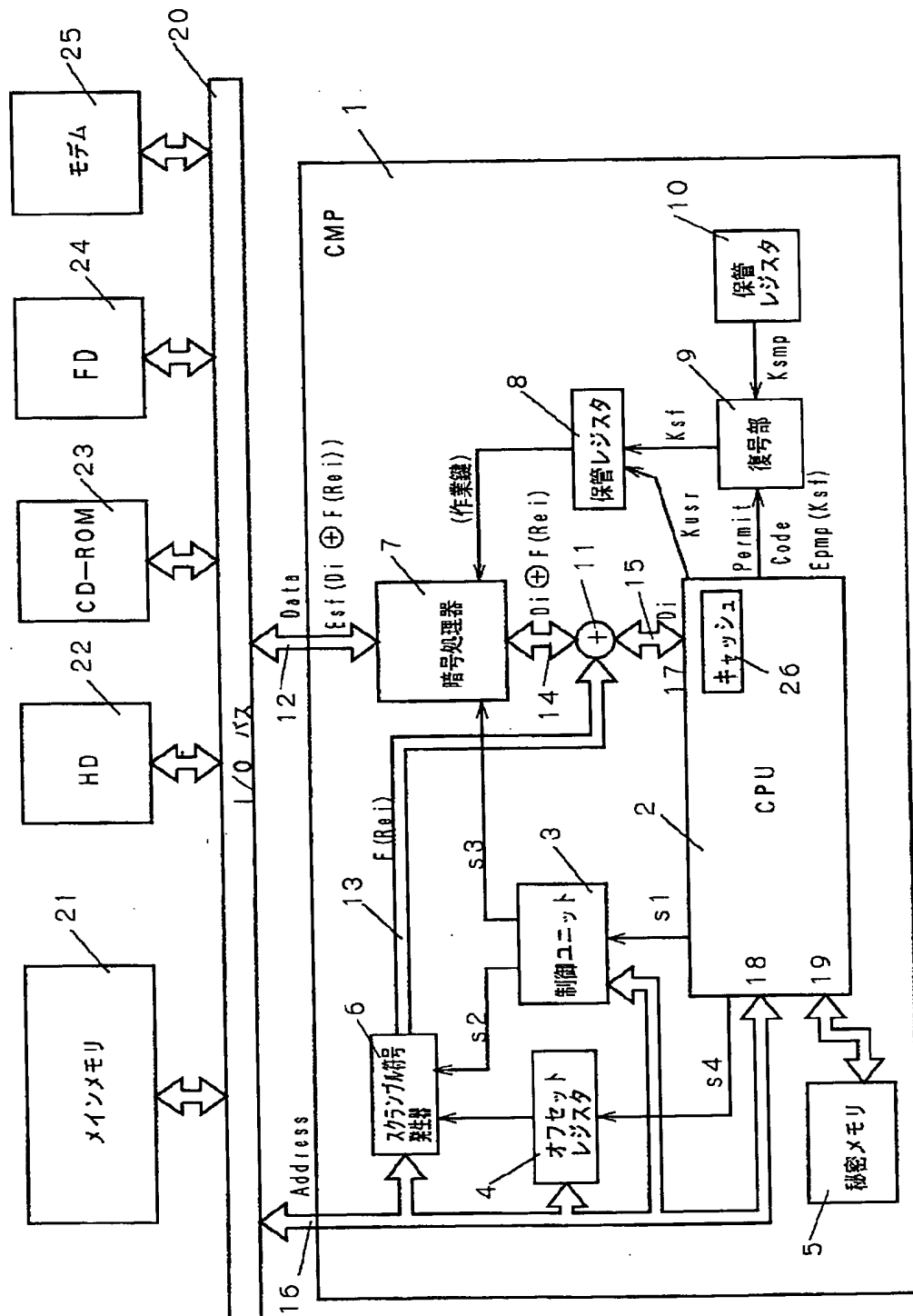
【 図16 】本発明の他の一実施例を示す構成図

【 図17 】本発明の動作説明に供するメインメモリの説明図

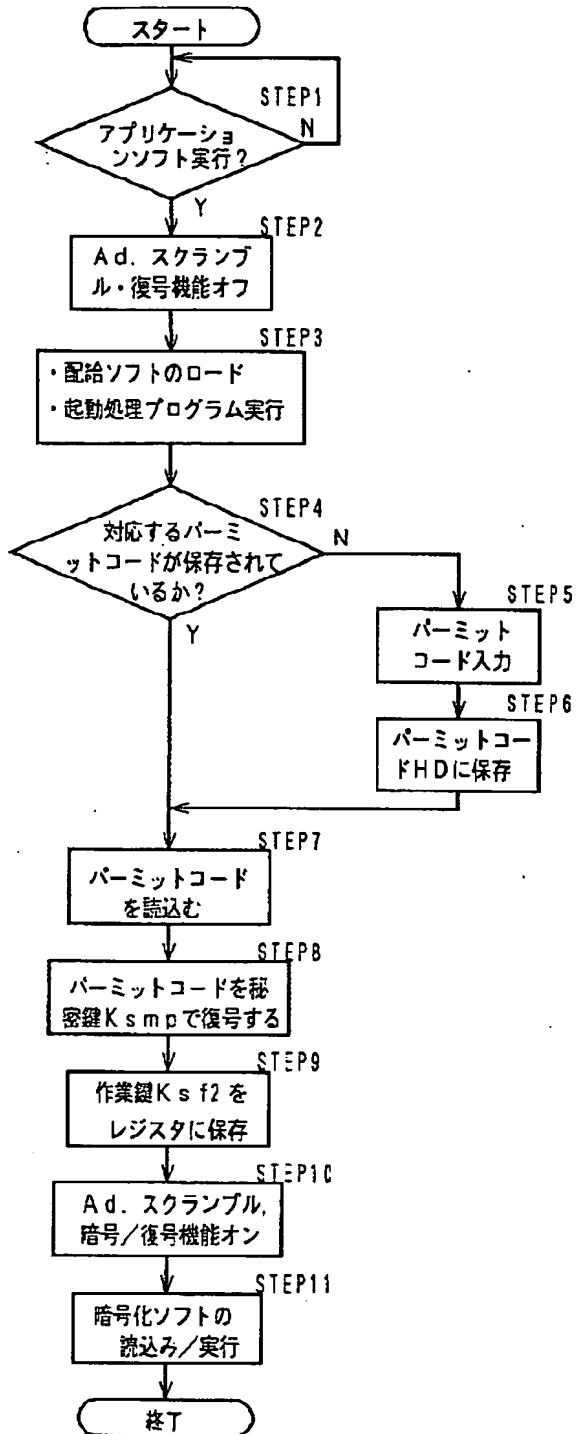
【 符号の説明】

- 1 CMP
- 2 CPU
- 3 制御ユニット
- 4 オフセットレジスタ
- 5 秘密メモリ
- 6 スクランブル符号発生器
- 7 暗号処理器
- 8 保管レジスタ
- 9 復号部
- 10 保管レジスタ
- 11 加算器
- 21 メインメモリ
- 22 ハードディスク
- 23 CD-ROM
- 24 フロッピディスク
- 25 モデム
- 26 キャッシュメモリ
- 30 CMP
- 40 CMP
- 41 暗号部
- 50 CMP
- 51 秘密メモリ部
- 52 照合手段
- 60 CMP
- 61 保管レジスタ部
- 70 CMP
- 71 多重鍵生成部

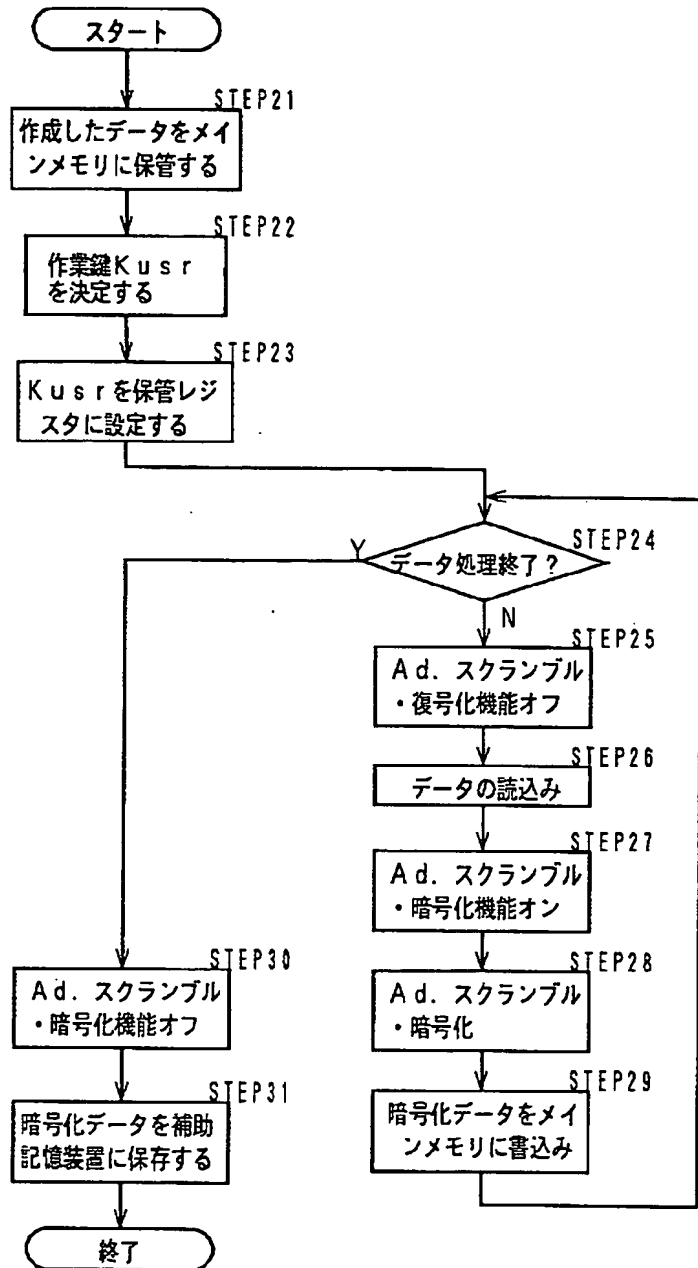
【 図1 】



【 図2 】



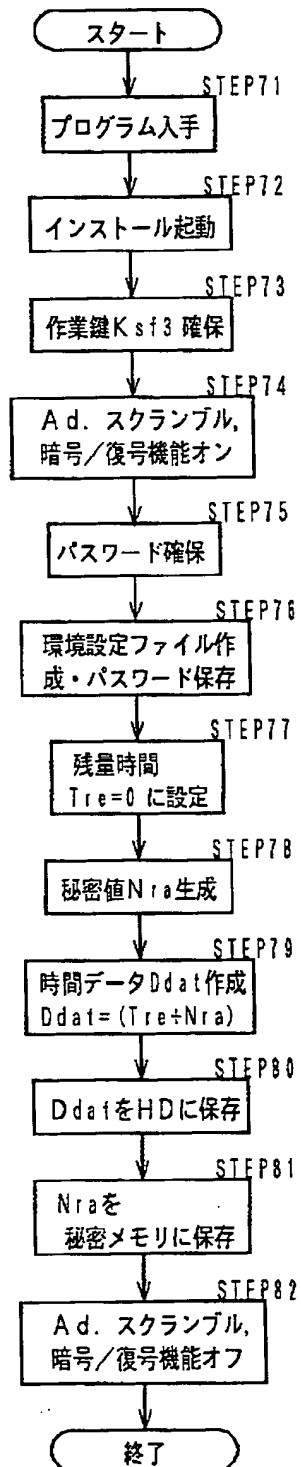
【 図3 】



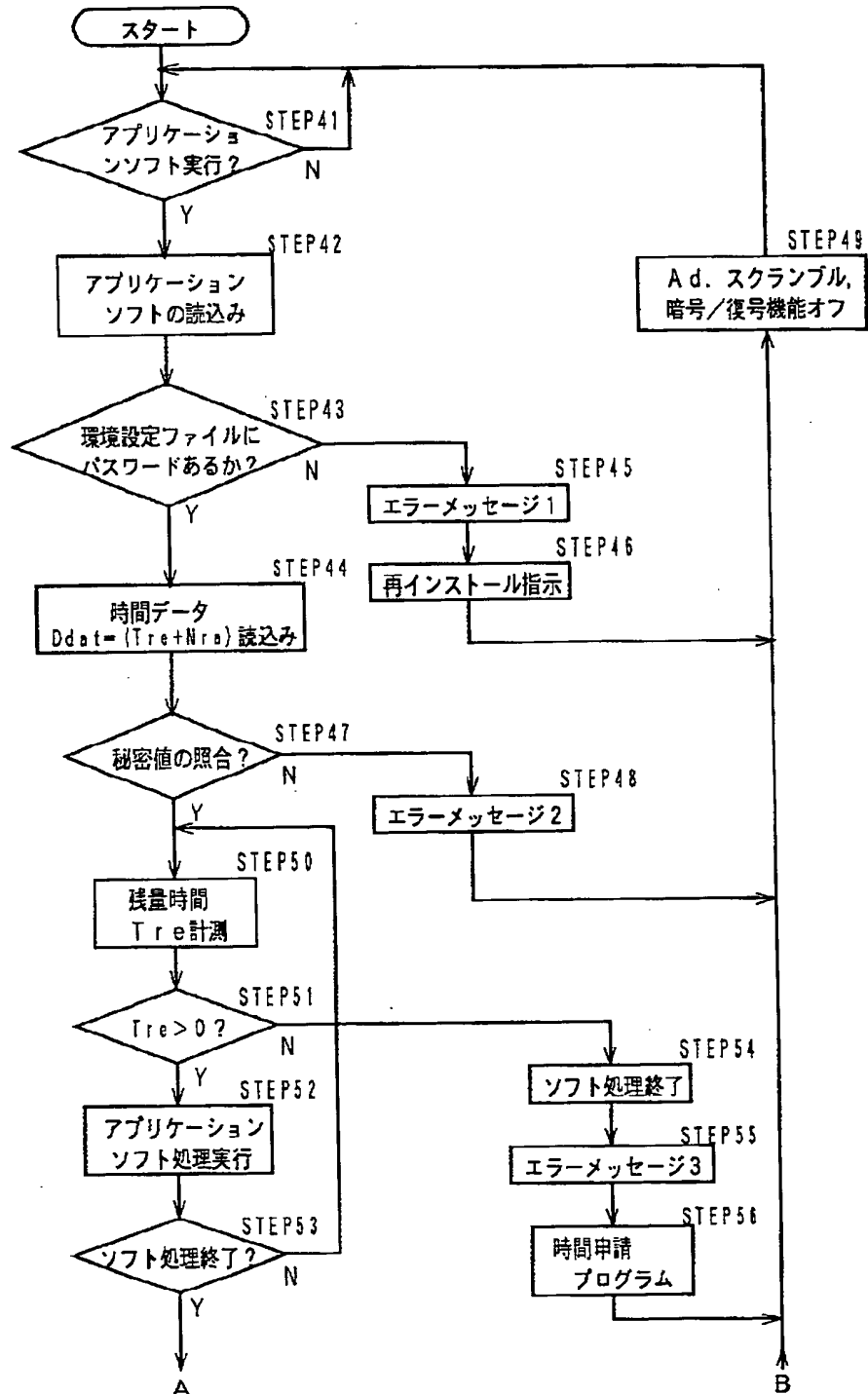
【 図10 】

起動処理プログラム	暗号化アプリケーションソフト Es f 2 (DapX ⊕ F (ReX))
-----------	---

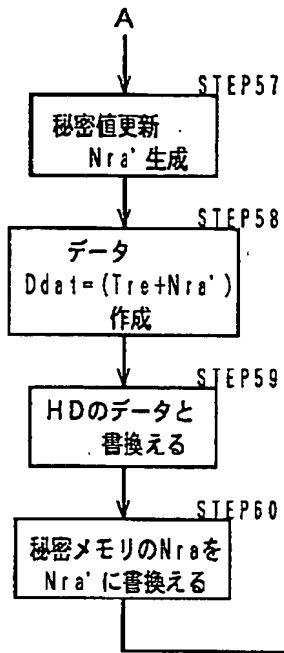
【 図4 】



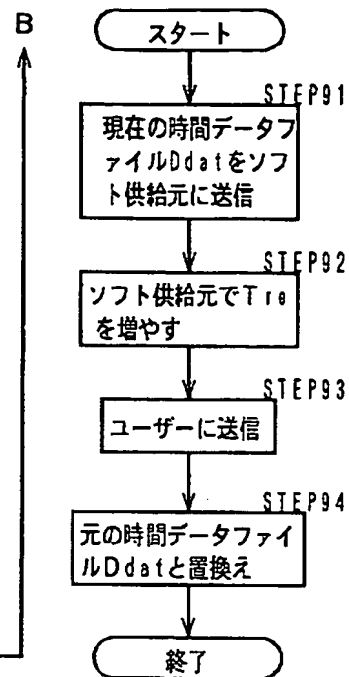
【 図5 】



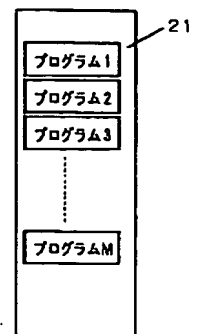
【 図6 】



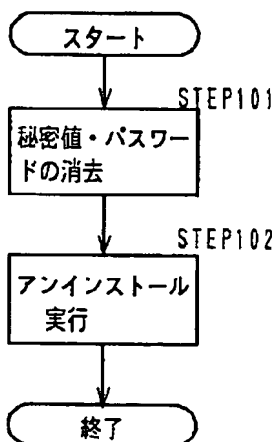
【 図7 】



【 図17 】



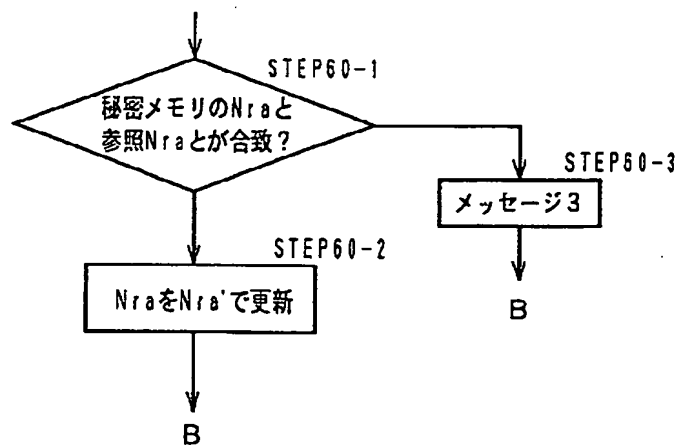
【 図8 】



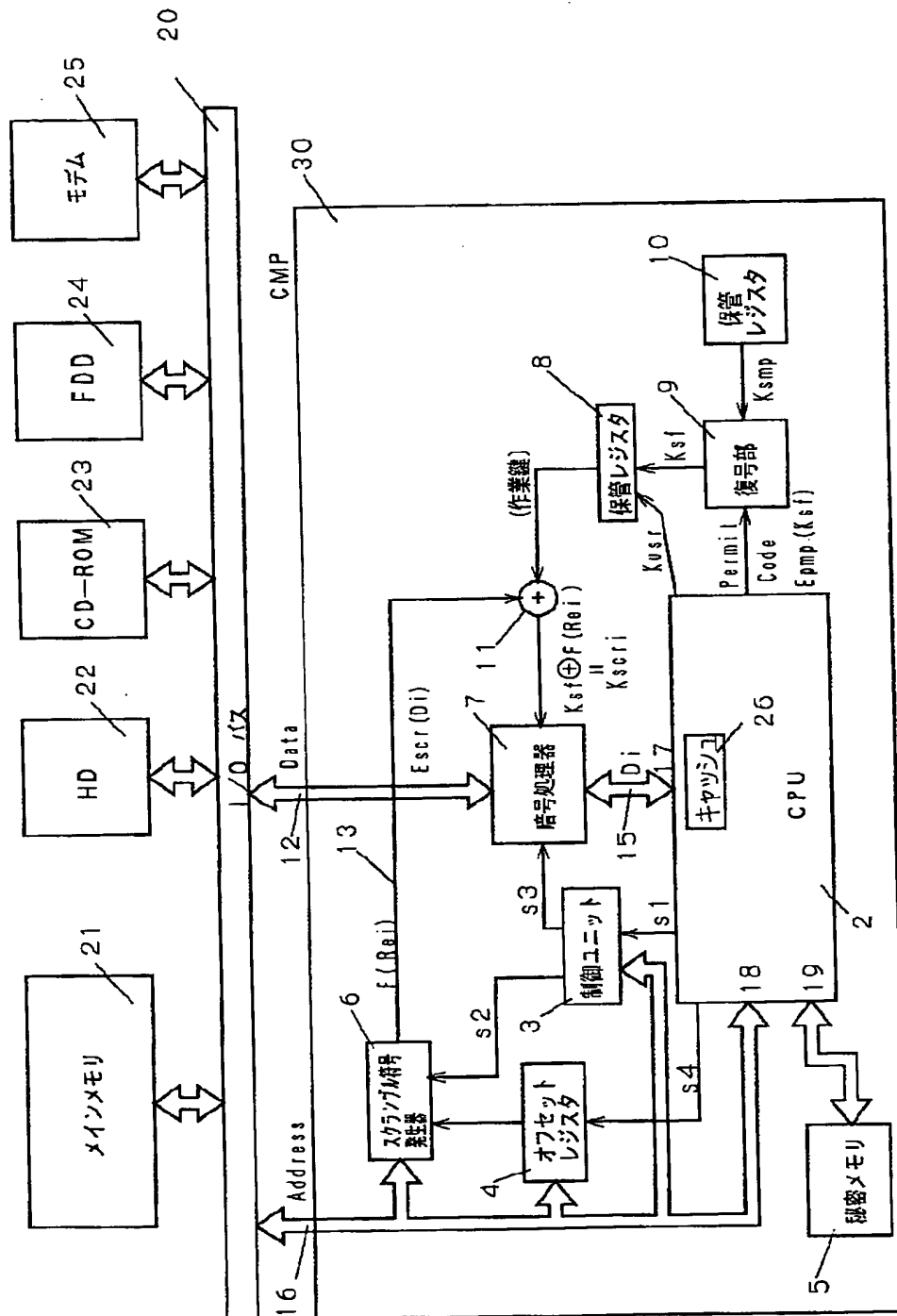
【 図9 】

パスワード	秘密値
PW1	Nra1
PW2	Nra2
PW3	Nra3
...	...
PW n	Nra n

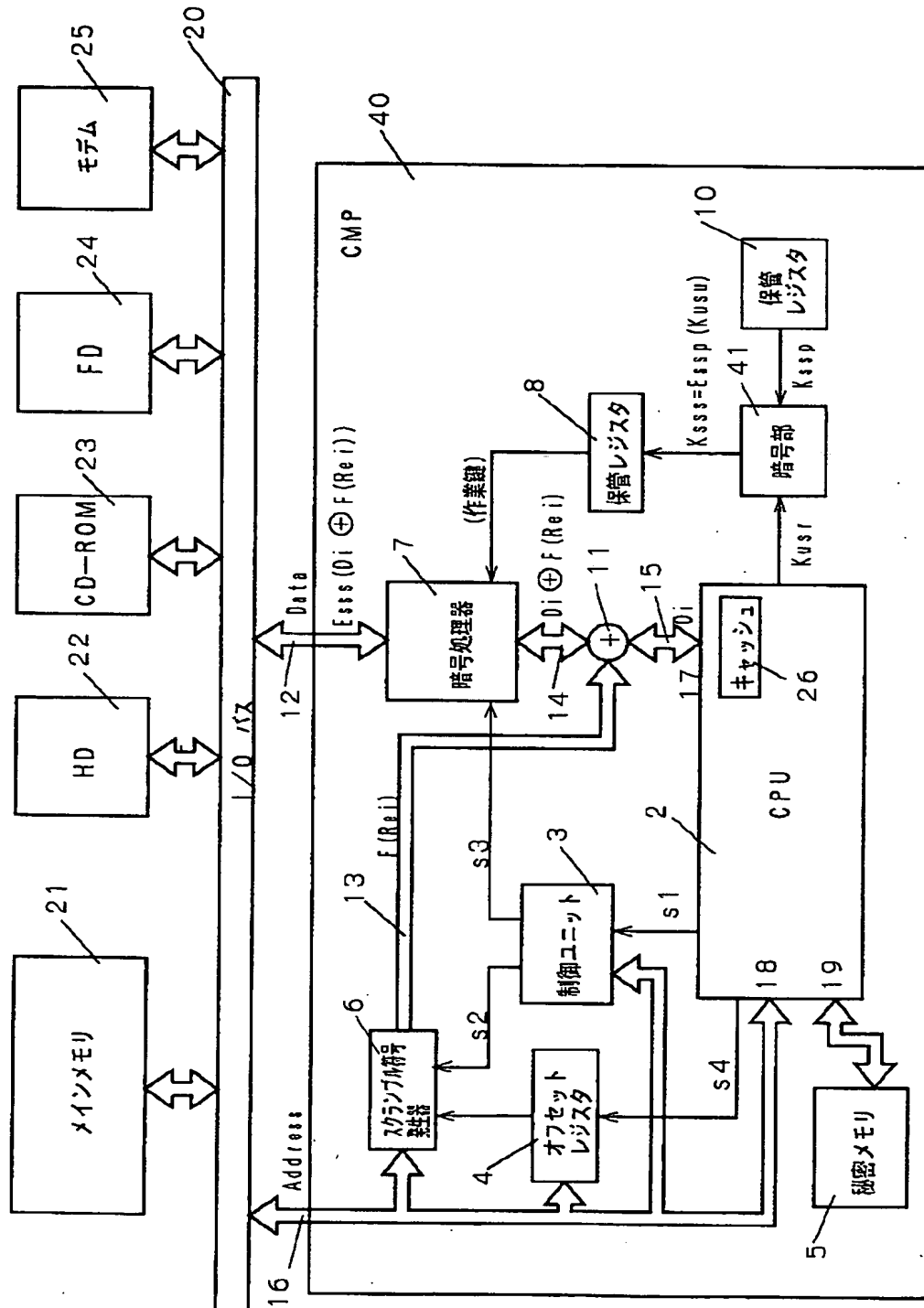
【 図11 】



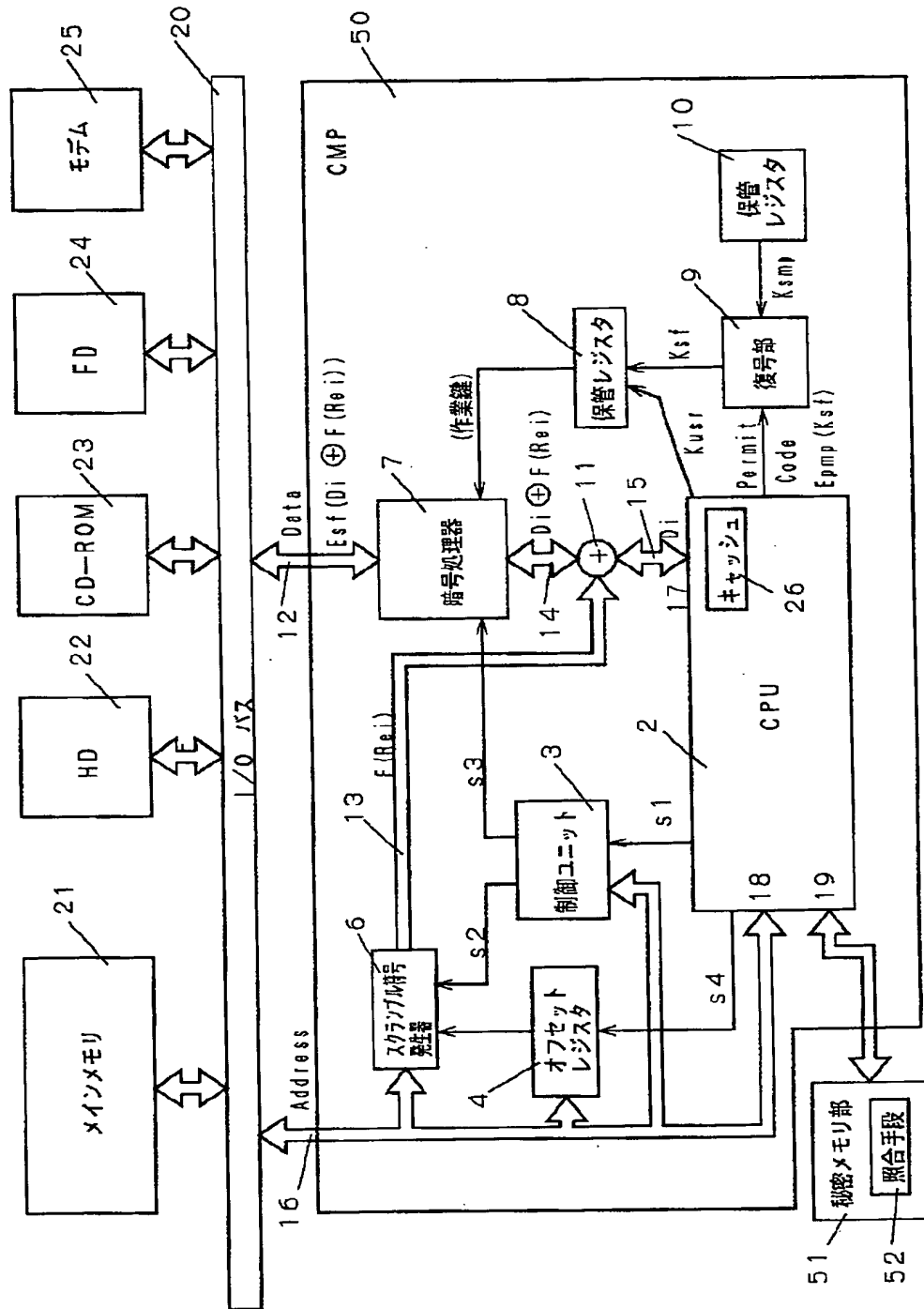
【 図1 2 】



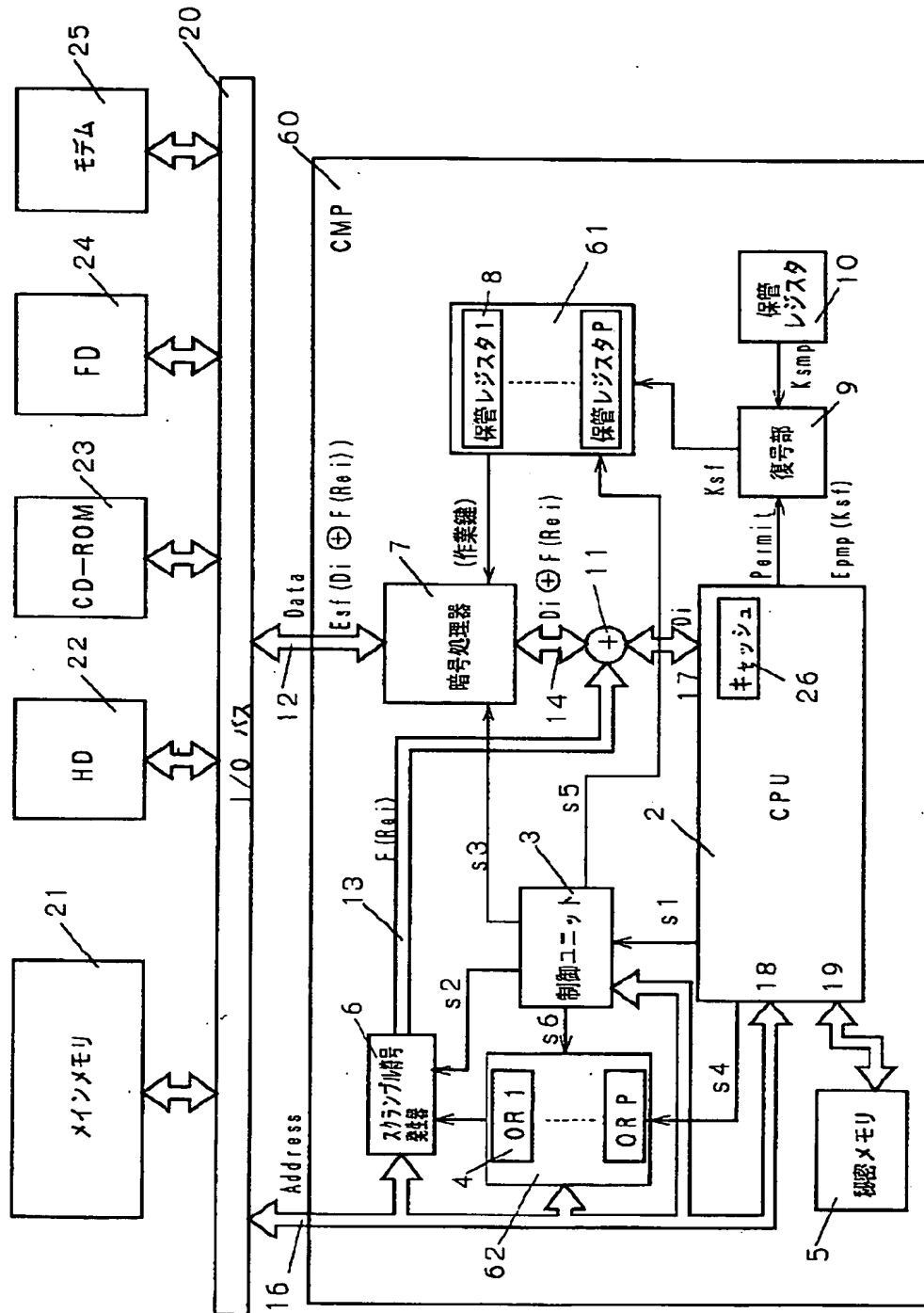
【 図13 】



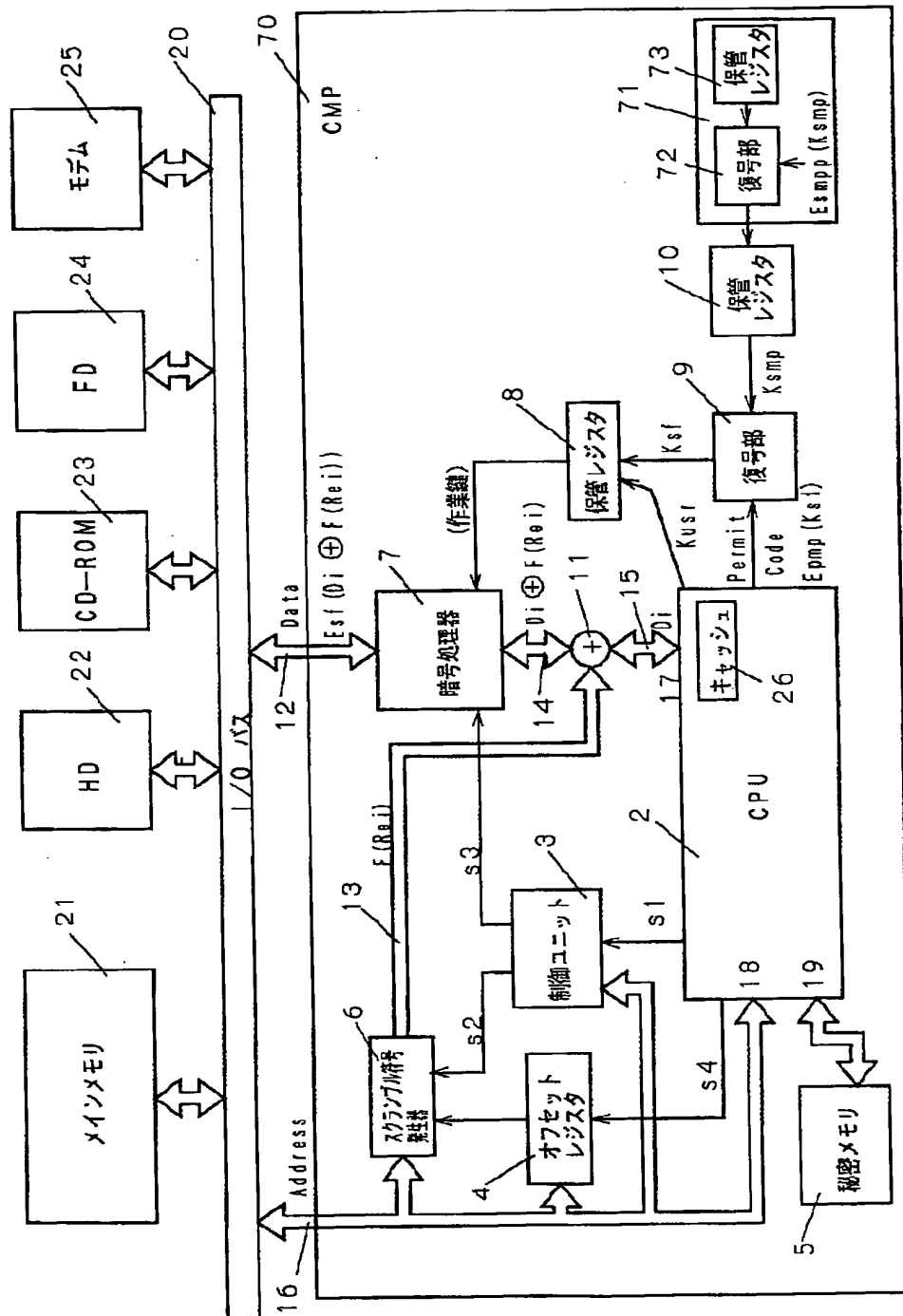
【 図14 】



【 図15 】



【 図16 】



フロントページの続き

(51) Int. Cl.<sup>6</sup>H04L 9/14  
9/32

識別記号

FI

H04L 9/00

641  
673A